Part

I

Overview of Commerce

COPYRIGHTED MATERIAL

In This Part

Chapter 1: Internet Era: E-Commerce
Chapter 2: Mobile Commerce
Chapter 3: The Important "-ilities"

CHAPTER 1

Internet Era: E-Commerce

This chapter does not intend to bore you with history and old-age content. Quite the contrary; we want to fast forward to new-age technology and e-commerce core concepts. However, it is essential to understand the basic yet prominent building blocks of the field of commerce before we dig into the new era. To grasp e-commerce, you need to understand the following concepts:

- Commerce
- Payment
- Distributed computing

Commerce and payment both have a much longer history than distributed computing, but that's the beauty of e-commerce; it is where the old world meets the new world! We are going to discuss how payment worked in the old days, and then describe how it operates now.

Evolution of Commerce

The Merriam Webster dictionary defines commerce this way:

1. social intercourse: interchange of ideas, opinions, or sentiments
2. the exchanges or buying and selling of commodities on a large scale involving transportation from place to place

With the recent popularity of digital social networking, the first definition of commerce is gaining more relevance; however, it is the second meaning that is our primary focus in this book[1]. We would also like to add the term "services" to "ideas" and "opinions" in the preceding definition so that the term becomes more relevant for our purposes.

Not only is commerce a fundamentally social phenomenon, it is also a very human-specific act. At its core, commerce is a kaleidoscopic collision of humans' unique ability to identify the need to optimize productivity, conserve energy, increase the chance of survival, exercise social abilities, and ultimately embark upon the act of exchange with other humans. Commerce is so deeply intertwined in our social fabric, and is such an integral part of our day-to-day life, it would be very hard to imagine civilization without it. By engaging in commerce, we express another facet of our intelligent and social behaviors. In other words, commerce is not just another simple human activity; it is a rather complex and sophisticated process that requires a great deal of knowledge, care, and attention to implement properly.

The oldest form of commerce is the barter system, which typically follows a direct-exchange mechanism where goods or services are directly exchanged for other goods or services. Barter is a direct system; a person offers goods or services to another person in exchange for goods or services that he needs. At its most basic form, the barter system suffers from scalability. That is, one has to physically carry the merchandise (in the case of goods), or be present personally (in the case of services) to interested parties, one by one, to be able to exchange for what he needs. Consequently, and to address this limitation, the *marketplace* was created — a place where merchants and customers show up during certain times and participate in exchanging goods and services. The marketplace is a social construct; that is, one needs to exercise communication, negotiation, and evaluation skills, among others, to successfully participate. The social facets of the marketplace are important here because they're also aspects of e-commerce. Examples include establishing trust, providing value for a purchase, facilitating delivery of goods or services, and many more.

Hard vs. Digital Goods

Before we proceed further with the foundations of commerce, it is important to note the differences between hard goods and digital goods. Early on, people identified value in two categories: tangible products and intangible products. As the name implies, tangible goods deal with the area of commerce that has to do with physical merchandise and products such as commodities, vehicles, devices, and so on. Intangible goods, on the other hand, include products that are not physical entities, such as insurance policies and refund guarantees for payments, and usually have more to do with services and promises of actions.

The concept of tangible vs. intangible goods is not specific to e-commerce; it has existed for almost as long as commerce has. Unlike most humans, computers work with binary values, zeros and ones, and digital entities. With the introduction of e-commerce, we have had to create yet another dichotomy to delineate what can be transported in computer-understandable format and what cannot. This is where the distinction between digital and hard goods is made. Computer-transportable products are referred to as digital goods, and all other products are hard goods (perhaps they are still resistant to becoming digitized).

Now we can have our very own definition of e-commerce as transporting any part of commercial tasks into the digital world so that computers can handle them. Seen from this perspective, then, it doesn't matter whether you are dealing with tangible products and hard goods online or going to brick-and-mortar stores, or whether you need to make a payment on your computer or walk into a banking branch. For example, it was not too long ago that if you wanted to have an album of your favorite rock star, you had to go to a store and buy the album in the vinyl or cassette formats. These were analog formats. Then with the advent of compact discs (CD) the music became digitized. The next step was for the Internet infrastructure to become more ubiquitous and offer higher bandwidths, and also for the computers to have proper programs to receive and play music tracks right on a user's personal computer. Once those requirements were satisfied, then the entire music delivery and consumption started to go online: hard goods gone digital. TV programs and other multimedia contents are following suit. In some parts of the world, you no longer need to buy a separate device (television) to watch your favorite programs; you can do all that on your personal computer.

The point is, the line between traditional and e-commerce is solid in some parts (for example you will always go to a dealer to purchase a car), but there are other parts of this line that are still blurred; you may at some point in the future have a hardware device on your personal computer that generates programmable odors in a digital form, so you won't need to buy perfumes anymore! The recent improvements in three-dimensional (3D) printing technologies, where you can actually print out 3D objects might be seen as a step in this direction. All that said, the objective of e-commerce is to take what was once part of tangible commerce and re-envision it for the digital world.

Payment

Payment is one of the cornerstones of any commercial activity, including e-commerce. At the end of a successful commercial exchange the buyer wants to receive his goods and the merchant her money. As humans engaged in commercial activities throughout history, the need to find creative ways to scale and expand it

became apparent. Introduction of money into commerce was a major leap toward making commerce scalable and to enable it to expand across the world. In this section, we discuss the foundation of payment, its main component (money), and the mechanics of money movement in modern systems.

Money

Early barter systems did not include a notion of money as it was a more advanced economic (and of course, social) construct that came in later. Money was invented to further facilitate commercial exchange. With the advent of money, humans were able to separate the notion of value from goods, represent it in an abstract form, and use it as an intermediary medium for commerce. Earliest forms of money were themselves material of intrinsic value (usually noble metals such as gold and silver), but the concept of it as an intermediary applies whether or not the medium has value itself. Money enables portability of value, scalability of exchange, and more novel governance and manipulation of value such as saving, investment, and other forms of economic growth. The scientific definition of money and its role in modern economy, in commerce, and in our social fabric are beyond the scope of this book, but suffice it to say that without money, civilization as we know it would most likely not exist.

Money is conceptually nothing other than a level of indirection to measure and represent value. Value demands management; therefore it would make perfect logical sense to assume that with the introduction of this concept (that is, money) came people and establishments that focused specifically to govern, manage, and handle it: the banks. Well, that's not exactly how it happened. In fact, the notion of a bank predates money. The first banks were probably the religious temples of the ancient world, and were probably established in the third millennium BC. Deposits initially consisted of grain and later other goods (including cattle, agricultural implements, and eventually precious metals such as gold, in the form of easy-to-carry compressed plates). Temples and palaces were the safest places to store gold as they were constantly attended and well built. As sacred places, temples presented an extra deterrent to would-be thieves. There are extant records of loans from the 18th century BC in Babylon that were made by temple priests/monks to merchants.[2]

Financial Networks

Money carried out commerce and humans' economic interactions for a couple of thousand years, but it had its limitations. For example, money wasn't a suitable system to manage credit (borrowing to spend, and paying at a later time) in a scalable way. Further innovations were needed to address such shortcomings, namely the introduction of credit and ways to manage value in forms suitable

for the digital age. The notion of credit as we use it in today's commerce evolved in the late 1960s. However, using a card to represent credit is a bit older.

The concept of using a card for purchases was described in 1887 by Edward Bellamy in his utopian novel *Looking Backward* (Signet Classics, 2000). Bellamy used the term "credit card" 11 times in this novel. The modern credit card was the successor of a variety of merchant credit schemes. It was first used in the 1920s, in the United States, specifically to sell fuel to a growing number of automobile owners. In 1938, several companies started to accept each other's cards. The concept of customers paying different merchants using the same card was implemented in 1950 by Ralph Schneider and Frank McNamara, founders of Diners Club, to consolidate multiple cards. Diners Club, which was created partially through a merger with Dine and Sign, produced the first "general purpose" charge card, and required the entire bill to be paid with each statement. That was followed by Carte Blanche and, in 1958, by American Express, which created a worldwide credit card network. However, until 1958, no one had been able to create a working *revolving credit* financial instrument issued by a third-party bank that was generally accepted by a large number of merchants. In September 1958, Bank of America launched the *BankAmericard* in Fresno, California. BankAmericard became the first successful, recognizably modern credit card, and with its overseas affiliates, eventually evolved into the Visa system. In 1966, the ancestor of MasterCard was born when a group of California banks established Master Charge to compete with BankAmericard; it received a significant boost when Citibank merged its proprietary *Everything Card* (launched in 1967) into *Master Charge* in 1969.[3]

A financial network is an immensely complex system. Credit Card, Insurance, Securities, and Banking are the main players of the financial services industry. Out of all the financial institutions, it is only the banks that are legally authorized to transfer the ownership of money. At a very high level, the banking business model is to borrow money at a low cost, lend it at a higher cost, and charge fees for moving the money from one account to another. There are many bank types: commercial, savings (for example, Cajas, Caixas, Sparkassen, and so on), building societies, credit unions, community banks, and so on. The two main categories of banking systems, however, are the wholesale (or commercial) and retail (or consumer) banking systems.

The rules, regulations, and operational aspects of wholesale banking are different than those of consumer banking. Traditionally banks deal with loans and deposits. Commercial banking loan and deposit operations typically deal with investment banking, equipment leasing and financing, commercial lending, line of credits (LOC), foreign transactions (ForeX), wire transfer, cash management, and commercial checking. On the other hand, consumer banking operations deal with auto loans, home equity lines of credit (HELOCs), credit cards, certificates of deposit (CD), and savings and checking accounts. This is illustrated in Figure 1-1.

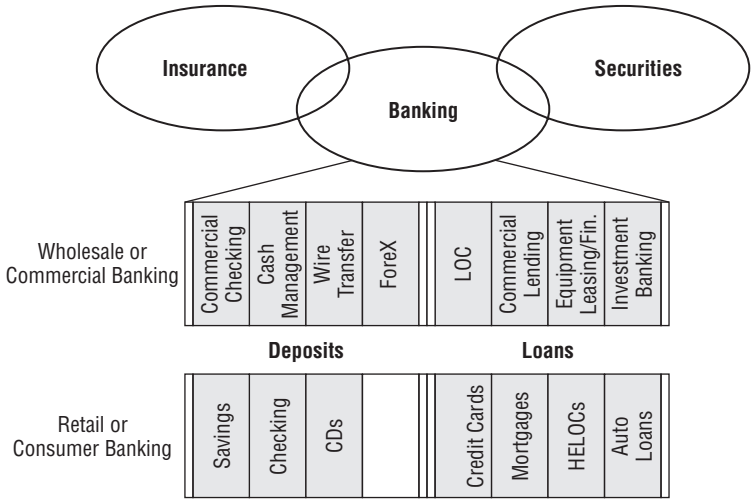

Figure 1-1: Financial services overview

At a very high level, a financial transaction takes place when a consumer (buyer) exchanges value with a merchant (seller). The buyer and seller are two of the main actors of both traditional commerce and e-commerce. The third actor is the financial institution (FI), which facilitates the value movement and charges the seller, the buyer, or both. All financial institutions (such as banks) operate within constructs called *payment networks*. As the term implies, a payment network is a networked system that facilitates the transfer of money (value) and cash-substitutes (prepaid cards, gift cards, and so on).

There are different types of payment networks; depending on the classification of financial institution, the nature of the transaction and the type of financial instrument (FI) used for the transaction, a different payment network is deployed. This is illustrated in Table 1-1.

Table 1-1: Payment Networks

FINANCIAL INSTRUMENT	NETWORK	EXAMPLES
Checks	Check clearing houses	
Credit cards	Credit card networks	Visa, MasterCard, Discover, AMEX
Debit cards	Debit card networks	STAR, LiNK, INTERLINK, Pulse, DK, NYCE, Electronic Cash, LASER, ACH (in the United States)
Direct debit/ credit	Clearing house network	

FINANCIAL INSTRUMENT	NETWORK	EXAMPLES
Wire transfer	Domestic wires	GSIT, CHAPS, Federal Reserve Financial Services
	International wires	SWIFT International
Private label credit	Private networks	Cetelem, GE, CitiBank
Prepaid cards	Private networks	Apple iTunes, AT&T, Starbucks, and so on

In Table 1-1, the two main networks, namely bank and credit card networks, deserve a little more attention as your e-commerce system will very likely deal with them very frequently.

ACH

Within the United States of America, a specialized electronic financial network, called Automated Clearing House (ACH) is used to facilitate interbank transactions. Here's how ACH works: To start an ACH transaction, the Receiver of the transaction (account holder) authorizes the sender (Originator) to issue an ACH debit or credit to an account. Banks identify their accounts by constructing numeric values that are the combination of a routing number and an account number. The combination of a routing number and an account number uniquely identifies an account to all the members of the financial network. An Originator could be a person or an entity such as a company. For legal reasons an ACH transaction, be it a debit or credit, cannot be initiated without a prior authorization by the Receiver. Once the authorization is given to the Originator by the Receiver, the Originator creates an ACH entry with its banking institution. This bank is called the Originating Depository Financial Institution, or ODFI. At this point, the ODFI sends the ACH entry to an intermediary entity, called the ACH operator (Clearing House), which then passes it to the Receiver's bank (Receiving Depository Financial Institution or RDFI). Depending on the type of transaction, the Receiver's account is issued a debit or a credit.

Regulations that govern the way the ACH network operates are established by the Electronic Payments Association (formerly known as the National Automated Clearing House Association, or NACHA) and the United States Federal Reserve. Per these regulations NACHA maintains the records of all ACH transactions. The way in which ACH makes money is simple: The Receiver pays nominal fees to the RDFI (and the Originator to ODFI) for its services. RDFI and ODFI also pay to both the Clearing House and NACHA for their services. The ACH process is illustrated in Figure 1-2.

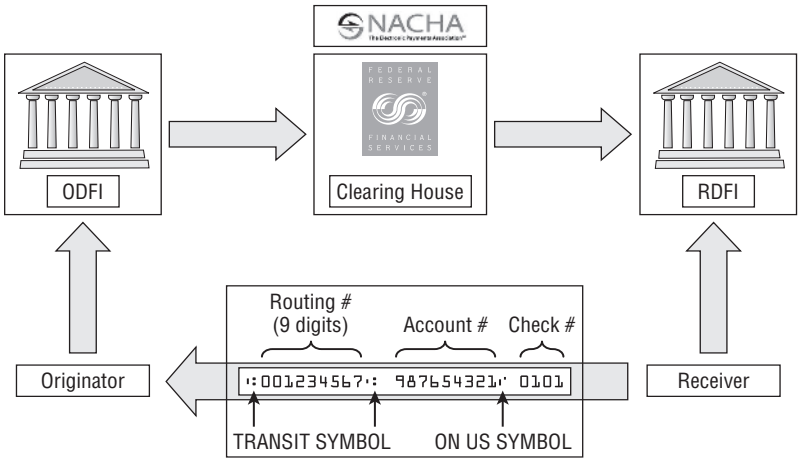

Figure 1-2: ACH process

Card Processing

ACH deals with the movement of money from one bank account to another. The ACH model, although specific to the United States, more or less has a similar foundation to most banking systems in other countries. The quintessential characteristic of an ACH transaction is that the money should exist in the creditor's account for the debit to take place. If no money exists at the time of transaction-commit, it fails. There are, however, cases where the bank charges a fee and agrees to commit the debit even though there are not enough funds to successfully commit the transaction; such cases are exceptional, usually incur an over-draft charge, and are not considered a usual operational model for bank accounts. The next and more recent model is the Credit Card system, also known as the Card Processing model, which as we discussed earlier, operates based on a borrow-first-pay-later business model. The card processing has two modes of operation: "four corner" and "closed loop." The four-corner model is used by Bankcards and Electronic Funds Transfer (EFT) networks, whereas the closed-loop model is used by private-label credit cards and stored-value cards.

The four-corner model has the following actors: Merchant, Acquirer, Issuer, and Cardholder. In the four-corner model, member banks act as issuers and provide credit cards to consumers. Acquirers are also member banks but act in a different capacity: They process transactions for merchants. The schemes (Visa, MasterCard, Star, and so on) set transaction rules, provide the processing switch (that is, the infrastructure that reroutes the financial transactions to the processing facility and card issuer), manage brand promotion, and most importantly, assist in performing risk management. The four-corner model distributes different roles to different actors.

In the closed-loop model, a single entity issues cards, handles merchants, sets transaction rules, provides switching services, and manages the network brand. The closed-loop model was originally created by American Express and is the operation model for Discover and JCB networks. Other closed-loop schemes include private-label credit cards and stored-value, and prepaid cards.

The full lifecycle of card processing, either four-corner or closed-loop, usually includes three phases:

- Authorization request
- Settlement
- Chargeback

Authorization request is a mandatory step and takes place when a transaction is initiated by the cardholder. If an authorization request fails, the transaction fails. The settlement phase is when the merchant attempts to settle all its charges against the network and initiates the actual money-movement process. The settlement phase is mandatory as well, and is the point where the merchant is also charged for using the card processing network services. The chargeback process is optional and occurs when a good or service is returned (or if fraudulent activities are performed against the cardholder's account) and the merchant has to credit the cardholder's account. Figure 1-3 illustrates the authorization request step of the four-corner card processing model.

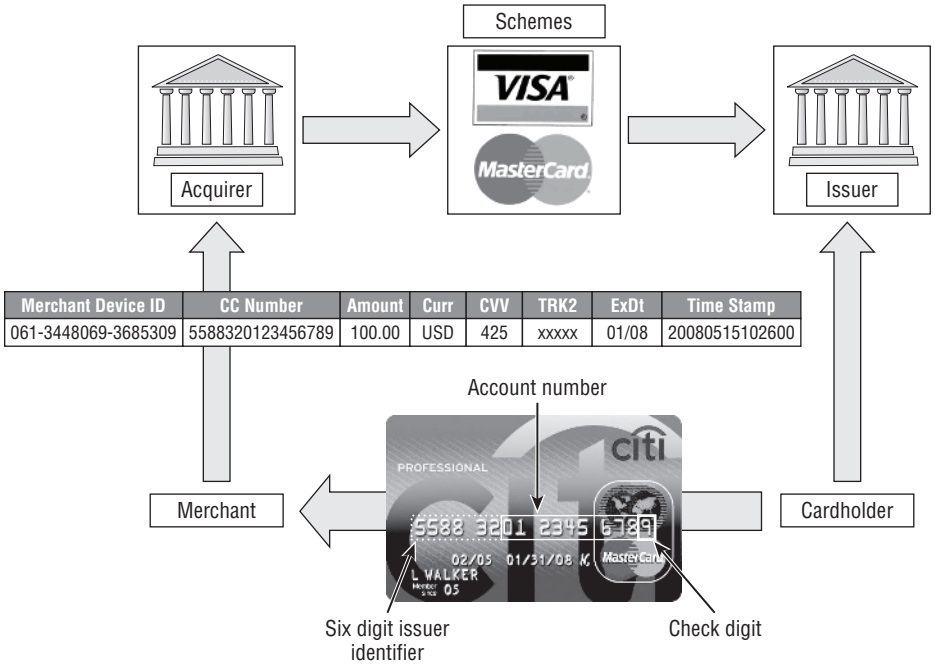

Figure 1-3: The four-corner model: authorization request

Credit card schemes operate based on a financial structure called the *interchange rate*. In layman's terms, the interchange rate is the cost of transferring money between the acquirer and the issuer. Interchange was originally intended to reimburse the issuers for some of their operational costs, and therefore is explicit revenue for the card issuers. As such, the interchange rate, although set by the card schemes, is not a revenue source for them. Interchange rates change periodically to adjust for the economic state of the regions where the networks operate. Furthermore, different types of transactions have different interchange rates; the motivation for this is to strike a balance between merchant acceptance and issuer incentives. These rates are approved by the respective scheme boards.

The process through which the rates are evaluated and decided is highly secretive; even the acquirers don't know the final rates and how they're calculated until the results are formally announced. Interchange rates are formally announced twice a year: in April and October. Figure 1-4 illustrates some of the factors that are known to impact the interchange rate calculation.

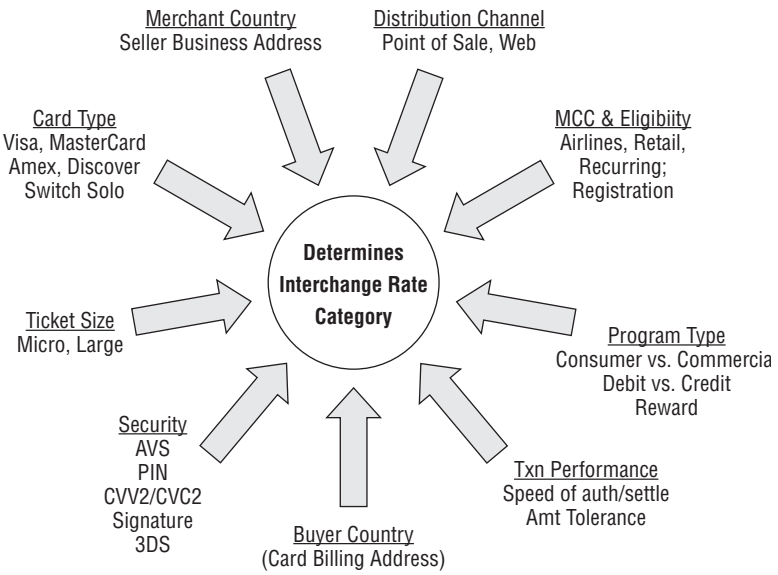

Figure 1-4: Parameters affecting interchange rates

The interchange rates in the United States are public information and are among the highest in the world. Other regions in the world do not officially publish their respective rates; however, after a short while the rates are known to the public. The reason that the interchange rates vary in different regions in the world is partly due to the fact that payment is a cultural phenomenon: Consumer payment preferences and purchase behaviors do not follow the same pattern from one culture to another. This difference in behavior consequently is reflected in each nation's banking system.

In the European Union, the Single Euro Payment Area (SEPA) delivers a single payment market that is used for all EU payments, both domestically and cross-border. Although SEPA offers a single currency, the payment models are fragmented when it comes to bank transfers and direct debits. Some countries in Europe are not even members of SEPA but rely heavily on commercial and financial interactions with SEPA members. The economic regions within the EU countries have different payment habits and banking behaviors. Some of these payment differences are highlighted in Figure 1-5.

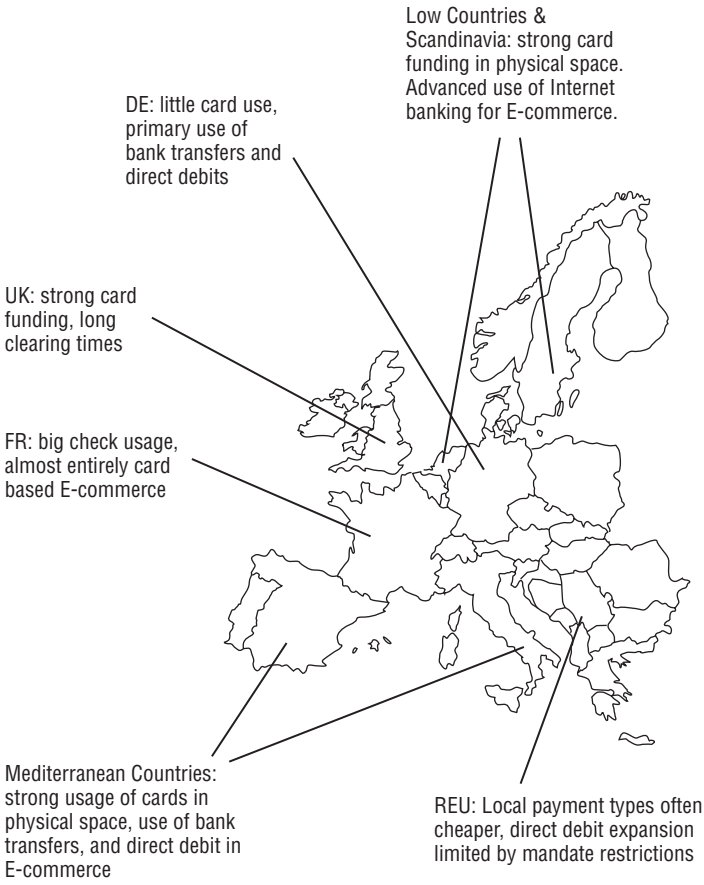
Low Countries & Scandinavia: strong card funding in physical space. Advanced use of Internet banking for E-commerce.
DE: little card use, primary use of bank transfers and direct debits
UK: strong card funding, long clearing times
FR: big check usage, almost entirely card based E-commerce
Mediterranean Countries: strong usage of cards in physical space, use of bank transfers, and direct debit in E-commerce
REU: Local payment types often cheaper, direct debit expansion limited by mandate restrictions

Figure 1-5: Different payment behaviors in the EU

The other end of the spectrum compared to the EU is the Asia-Pacific (APAC) region, where 45 percent of the world's population exists in only two countries (China and India) and is becoming an emerging and immensely powerful financial powerhouse. The APAC countries are usually highly regulated financial environments, with a great appetite for travel, gaming, and mobile payment solutions. The countries in the APAC region tend to move away from traditional

(hard-currency) payments and toward electronic payment to increase tax revenues and reduce the financial risks imposed by black markets. Figure 1-6 illustrates the characteristic financial behaviors of some main APAC countries.

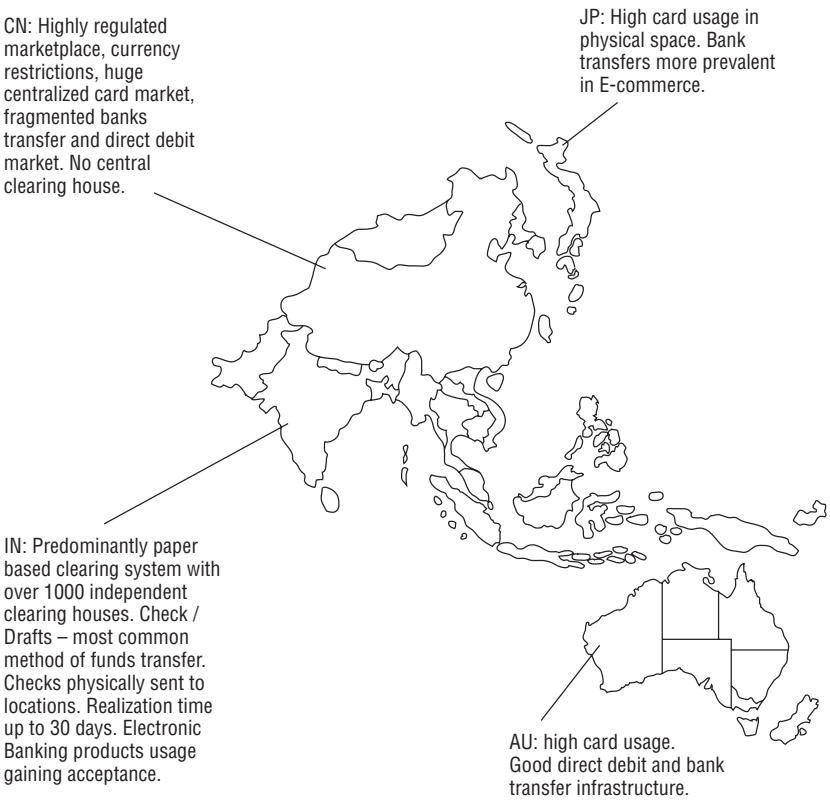
CN: Highly regulated marketplace, currency restrictions, huge centralized card market, fragmented banks transfer and direct debit market. No central clearing house.
JP: High card usage in physical space. Bank transfers more prevalent in E-commerce.
IN: Predominantly paper based clearing system with over 1000 independent clearing houses. Check / Drafts – most common method of funds transfer. Checks physically sent to locations. Realization time up to 30 days. Electronic Banking products usage gaining acceptance.
AU: high card usage. Good direct debit and bank transfer infrastructure.

Figure 1-6: APAC financial behaviors

There are also Latin American (LatAm) countries where modern financial behaviors are making headway, but cross-border transactions are typically very cumbersome and restrictive. Even in some of the more emerging countries in the LatAm region, namely Brazil and Argentina, the number of acquirers is very limited (in the single digits) and local bank withdrawals are typically allowed only for permanent residents and citizens. All of this is to drive a single point home: The dissimilarity in payment and banking behaviors not only affects the payment mechanisms and their corresponding infrastructures, but also anything that relies on these financial institutions, most notably e-commerce solutions.

Mobile Payment and Commerce

A study of payment technologies cannot be deemed complete without touching upon new payment mechanisms, chief among them the emerging M-Payment

and M-Commerce (mobile payment and commerce respectively). In short, mobile payment uses the mobile phone (or other mobile device) as a conduit to access an account (the purchasing account). A purchasing account could be the credit line (that is, the spending limit for a certain period of time such as $20 per month) that is made available to the mobile subscriber by the phone carrier (called carrier-billing), it could be a stored-value account (such as PayPal), or it could be the credit card or even a regular bankcard or bank account that is settled through ACH. There are different flavors of mobile payment and the field enjoys a plethora of new ideas that are emerging as a result of smart phones and mobile devices that are capable of running third-party applications. Mobile payment solutions potentially provide consumers with enhanced levels of safety and security of their financial information, as well as convenience. For merchants, on the other hand, mobile payments can mean lower fees, an increase in sales, and a higher financial bottom-line. The carrier-billing model can be an additional source of revenue for mobile network operators with great potential for realizing profits. Mobile payment is a very promising emerging field in financial industry and all the figures indicate a tremendous growth in its adoption. Because some of its models do not require the user to have a bank account, mobile payment is considered one of the most prominent solutions to penetrate regions in the world where banking infrastructure is weak or non-existent.

One of the issues with mobile payments in the market today is the lack of a clear and shared definition across the industry. There is often confusion and overlap between a mobile payment, mobile banking, and the use of the mobile phone to simply order goods or receive delivery (while paying by other means).

There are five main methods for using mobile phone financial transactions. These are:

- **Mobile order:** Transactions where the mobile phone is used simply to initiate the order, but not to make the payment.
- **Mobile payment:** A payment (transfer of funds in return for a good or service) where the mobile phone is involved in the initiation and confirmation of the payment. The location of the payer is not important: he may or may not be "mobile" or "on the move" or at a Point of Sale (PoS).
- **Mobile delivery:** Transactions where the mobile phone is used simply to receive delivery of goods or services (for example, an event ticket) but not to make the payment.
- **Mobile authentication:** Use of the mobile device to authenticate the user either as part of a payment transaction or to give access to some information or functionality.
- **Mobile banking:** Access to banking functionality (query and transaction) via the mobile phone. This means that the same (or a subset of) banking functionality that is available on the Internet is now available via mobile phones.

As you noticed, we don't classify initiating a payment (such as a bank transfer while within the mobile banking provided by banks) as a mobile payment: It is simply a feature of mobile banking.[4]

In addition to mobile payment options, other payment solutions are gaining traction in the financial industry. A very active solution is iTunes by Apple, Inc., which is categorized as a micropayment aggregation mechanism. Apple iTunes is considered a micropayment solution because it is still limited to digital content purchases such as music, video, digital books, and so on with a low value. Similar to other emerging payment solutions (for example, mobile), Apple iTunes will become a serious contender in financial space when it will also be used to purchase big-ticket items.

Another new-wave payment solution is to use contactless devices for payment. Contactless (also known as proximity, vicinity, or NFC) technologies have gained their name because, unlike the traditional plastic cards, they do not rely on a physical contact with the PoS to initiate transactions. These devices usually include an inductive antenna, which is energized by way of being close to an electromagnetic field (provided by the PoS), and a microchip that contains financial information of the holder. Almost all major financial institutions have a live contactless project in some parts of their network, or are working on one. Examples include payWave by Visa, PayPass by MasterCard, ExpressPay by AMEX, and Zip by Discover networks. Albeit a very convenient payment technology, a major impediment in the adoption of contactless payment solutions has been the need for modifying the PoS. In other words, the PoS device must be changed to be able to support contactless devices.

In this section, we have covered only the very basics of the payment industry: merely the first couple of letters in the payment alphabets. Other advanced topics such as large-value payments (the process where, for instance, a state buys aircraft from another state), commercial payments, general purpose payment cards (GPPC), the details of EFT- and PIN-based networks, PIN-less debit transactions, funding costs and managing transaction expenses, inner-workings of global payments, cross-border payments and their tax ramifications, and emerging payment technologies (such as contactless and mobile payment, carrier-billing, and near-field communication or NFC-based instruments) are all hot and current issues in the payment industry and each deserves a book to do it justice.

Distributed Computing: Adding E to Commerce

We have covered the basics of commerce and payment. Combining the two with a little bit of high-tech and distributed computing will get you exactly what you want: an operational, functional, scalable, and secure e-commerce infrastructure. In this section, we describe the technical foundation and key elements of distributed computing and how they contributed to the advent of

online commerce. Let's start with the client/server model and make our way up to the more advanced cloud computing paradigm because that is how scalable e-commerce sites have started and ended up. This section is more technical in nature so fasten your technical seatbelts as the fun is about to start.

Client/Server

In the first chapter, we covered the basics of computing and noted its primary model: standalone computing in a single system. The standalone computing paradigm is usually monolithic. That is, a combination of all the steps necessary to complete a task is put in a program and once you run the program it handles the tasks in one piece. The next step in the computing paradigm is to identify generic tasks and devise components that are responsible for handling only those specific tasks. One model of achieving this goal is to divide the tasks into client and server tasks respectively. A client/server computing paradigm usually has the following constructs:

- **Service:** Client/server is primarily a relationship between processes running on separate machines. The server process is a provider of services. The client is a consumer of services. In essence, client/server provides a clean separation of function based on the idea of service.
- **Shared resources:** A server can service many clients at the same time and regulate their access to shared resources.
- **Asymmetrical protocols:** There is a many-to-one relationship between clients and server. Clients always *initiate* the dialog by requesting a service. Servers are passively waiting on requests from the clients.
- **Transparency of location:** The server is a process that can reside on the same machine as the client or on a different machine across a network. Client/server software usually masks the location of the server from the clients by redirecting the service calls when needed. A program can be a client, a server, or both.
- **Mix and match:** The ideal client/server software is independent of hardware or operating system software platforms. You should be able to mix and match client and server platforms.
- **Message-based exchanges:** Clients and servers are loosely coupled systems that interact through a message-passing mechanism. The message is the delivery mechanism for the service requests and replies.
- **Encapsulation of services:** The server is a "specialist." A message tells a server what service is requested; it is then up to the server to determine how to get the job done. Servers can be upgraded without affecting the clients as long as the message interface is not changed.

- **Scalability:** Client/server systems can be scaled horizontally or vertically. Horizontal scaling means adding or removing client workstations with only a slight performance impact. Vertical scaling means migrating to a larger and faster server machine or multi-servers.
- **Integrity:** The server code and server data is centrally maintained, which results in cheaper maintenance and guarding of shared data integrity. At the same time, the clients remain personal and independent.[5]

The preceding characteristics suggest a very simple model to describe the client/server paradigm; by defining generic services and standard interfaces to access those services, the client/server paradigm creates an intermediary layer, namely middleware, to demarcate client tasks from those of the server. This is depicted in Figure 1-7.

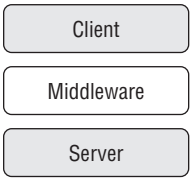

Figure 1-7: Components of the client/server paradigm

The client/server computing paradigm is the simplest form and one of the earliest models to implement distributed computing. The client/server model is simple; however, by no means is it "easy" to implement. An efficient and successful implementation of a client/server model requires a great deal of expertise and experience. This basic computing paradigm, albeit following a simple concept, is a very powerful design pattern and should not be dismissed as an "old concept"; there are many tasks and complex functionalities that are still best accomplished through the client/server design model. Examples include database query tasks, atomic and stateless operations (i.e. operations that do not persist data from one invocation to the next), and many more basic yet powerful design tools.

Grid Computing

Grid computing and its later child, cloud computing, were conceived to build on top of the client/server computing paradigm to address its major shortcoming: scalability. One of the earliest proponents of grid technology is Ian Foster of the Argonne National Laboratory and professor of Computer Science at the University of Chicago. In 1998, in a book called *The Grid: Blueprint for a New Computing Infrastructure* co-authored with Carl Kesselman, Foster defined the grid as "a hardware and software infrastructure that provides dependable,

consistent, pervasive, and inexpensive access to high-end computational capabilities." Over the years, even Foster's definition of a computational grid has evolved, by his own admission. In a subsequent article, *The Anatomy of the Grid,* co-authored with Steve Tuecke in 2000, he changed the definition to include some element of social and policy issues, stating that grid computing is concerned with, "coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations."[6]

Grid computing is a distributed computing design pattern that is usually geared toward solving a specific problem, such as payment, identity services, scalable IP (Internet Protocol) telephony, and so on. Although grid computing has become the buzzword in both industry and academic communities, it is not a technology that has been developed from scratch. Rather, it is a conglomeration of different existing technologies such as cluster computing, peer-to-peer (P2P), and Web services technologies.

During the last decade, different technology elements such as cluster and P2P computing have evolved from the distributed and high performance computing communities, respectively. In cluster computing, different computing resources such as machines, servers, and so on are connected by high-speed inter-connects such as Infiniband, Gigabit Ethernet, and so on to provide high performance. Computing paradigms such as Message Passing Interface (MPI) and Parallel Virtual Machines (PVM) allow programmers to write parallel programs for clusters. A P2P system, on the other hand, allows peers or computers to share resources. They are suitable for storing files or information either in an unstructured or a structured P2P mode. YouSendIt[7] is a classic example of unstructured P2P where users store the files and a particular request is processed in a heartbeat. Structured P2P, on the other hand, uses structures such as mesh or ring, more generically called the Distributed Hash Table (DHT), so that the search time for information retrieval is bounded. CHORD[8] and CAN[9] are examples of structured P2P systems that are based on the principles of the distributed hash table.

It would be unfair to say that the high-performance computing community solely contributed to the development of clusters and the distributed computing community's work resulted in the development and later flourishing of the P2P systems. There was a fair amount of technical interaction between these two different communities resulting in the final evolution of P2P and clusters. Similarly, these two different technologies contributed a lot to the eventual acceptance of grid computing as a promising IT virtualization technology. In terms of concepts, grid computing combines the unique points of both P2P and clusters. Out of many participating technologies, the Web services are an integral part of grid computing and have resulted in the further evolution of distributed computing into grids, and later on to clouds. Figure 1-8 illustrates an abstract evolution of the grid computing from P2P and clusters, to the introduction of Web services technologies, and finally to clouds.

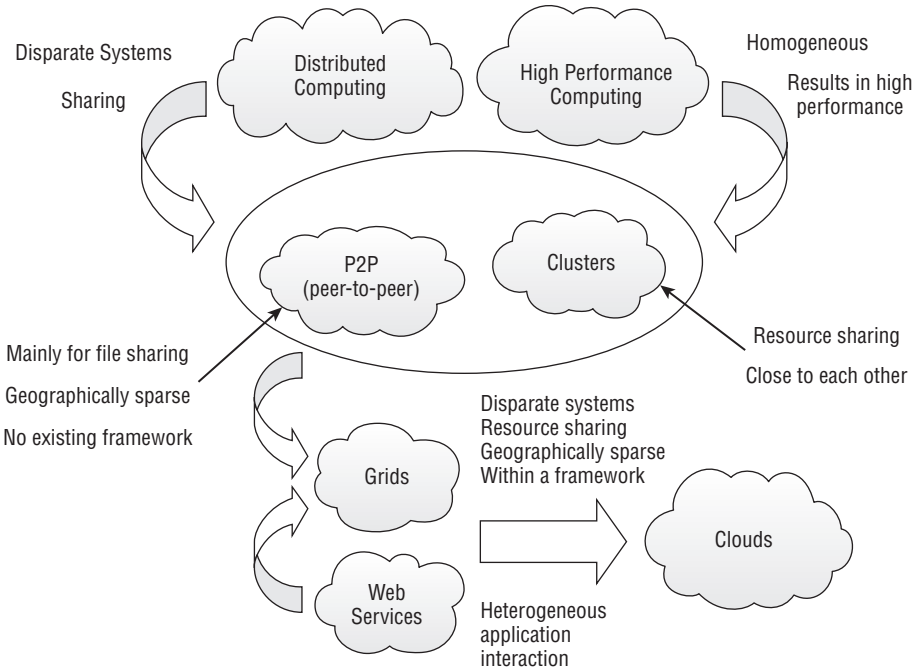

Figure 1-8: Evolution of Grid Computing

Cloud Computing

Cloud computing is a natural evolution of the grid computing paradigm, which approaches distributed computing from a slightly different perspective. Cloud computing is a computing paradigm that combines architectural elements (for example, virtualization or Web services) with business models (for example, subscription-based and pay-per-use) to deliver IT services across the Internet. Because the majority of cloud computing business models rely on maximizing profit and utilizing the computing infrastructure, the infrastructure must be very efficient. To be efficient, the infrastructure needs to expose, at the very least, the following properties:

- High utilization
- Dynamicity — fast turnaround to limit idle capacity
- Automation — lower management costs

Implementing the cloud computing business model would also require specific capabilities such as accounting and chargeback, as well as service cataloging (enabling self-service capabilities for its users).

In the same way applications have been deconstructed to evolve from monolithic to distributed, to service oriented architecture (SOA)-based models, cloud

computing deconstructs the datacenter by distributing its functionalities across the Internet and making those functionalities accessible as services. Following the same evolution, the datacenter functionalities in question are evolving from custom built and special purpose, to become services that are shared among various users, potentially external to the legacy environment.

In any case, these components are still implemented in real datacenters, with real hardware and real system administrators, and cloud computing (or the functions offered by cloud computing) can be seen as a layer on top of an existing infrastructure, so long as the infrastructure satisfied the requirements of said services.

In the specific case of cloud computing, applications are deployed on top of this infrastructure, interacting with the infrastructure services, either locally or through the Internet. Optionally, applications can interact with the cloud management services directly if they need to drive real time infrastructure changes (for example, an application can request resources itself). This is depicted in Figure 1-9.

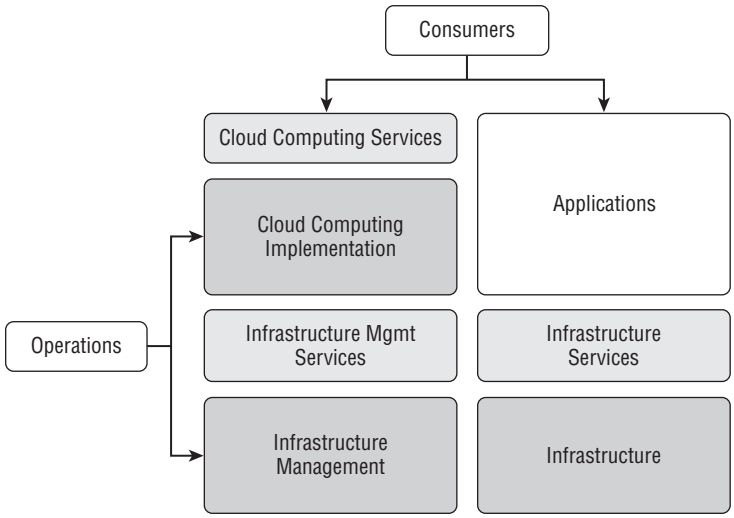

Figure 1-9: Cloud computing high-level architecture

Below is the description of cloud computing main components that are shown in Figure 2-9, along with their responsibilities:

- **Cloud computing services:** Services exposed by the cloud computing implementation. This can be platform as a service (PaaS), storage as a service (SaaS), software as a service (SaaS), and so on. These services are exposed either through a self-service user interface (UI) or a programmatic interface.
- **Cloud computing implementation:** A set of components that implement the cloud computing services. This includes resource management, resource allocation and optimization, account management and chargeback,

accounting and monitoring, user interface, service backend, and so on. These components are collectively layered on top of infrastructure management functions and access them through a service layer.

- **Infrastructure management services:** This is a service layer that exposes the functions of the infrastructure management. This may be built specifically for the purpose of the cloud computing implementation, or could exist only to support other use cases such as process automation.
- **Infrastructure management:** A set of components that implement the infrastructure management functions. This includes servers, hypervisors, network, storage management, platform management, application management, monitoring, infrastructure service management, and so on.
- **Infrastructure:** The infrastructure layer is composed of server and network storage resources, implementation of infrastructure-specific services such as log rotation services, firewall services, and key management.
- **Infrastructure services:** In this context the infrastructure services are the services that are required by the applications or other services deployed on the infrastructure.

Cloud computing is the foundation of scalable e-commerce systems. The following sections delve deeper into the technical and operational details of cloud computing, its main characteristics, and how they are implemented.

Shared Resources

One of the key requirements to increase the utility of the infrastructure is the capability to share resources across multiple workloads. In the most favorable scenario, the pattern of resource usage should be a non-overlapping distribution model that allows for the greatest infrastructure utilization. The strategies to distribute workloads on shared resources are an important aspect of maximizing the infrastructure utilization.

Dynamic Resource Allocation

Traditional resource allocation is static, using the estimated peak capacity as the basis to calculate the allocated resources. Cloud computing allows the resources to be allocated as needed and as the demand increases, or de-allocated as demand decreases. The infrastructure should determine when and which resources should be added to specific applications. Dynamic resource allocation has the following components:

- The definition of a global pool, from which all resource requests are satisfied, allowing global resource optimization

- The capability to match resource requests to available resources in accordance to policies and business objectives
- The ability to reject resource requests in a graceful manner based on policy exceptions
- The capability to predict and plan resource utilization
- Policies to drive the resource sharing behavior
- Optimization of resource allocation based on usage and Service Level Agreements (SLA)

Physical Abstraction

There should be no dependencies between the application and an infrastructure implementation that would prevent relocation of the application to other locations in the cloud. This abstraction is critical for resources such as IP addresses, data sources, and file systems. Physical abstraction is accomplished by satisfying the following additional requirements:

- Virtualization of the hardware.
- Designing applications with no local dependencies (local hard disk, specific hardware, hardwired resources such as IP addresses, and so on).
- Late binding between the services and their physical realization. That is, all such bindings are performed during the runtime of the system.

Utility Model

One of the characteristics of the utility model is a pay-as-you-go chargeback. While this might not be a goal at your organization, accounting of usage based on actual allocation is an incentive to keep the resources low in the beginning and to optimize them over time. This model will allow your e-commerce cloud to scale gracefully.

Self Service

One of the key goals of cloud computing is to enable the fastest time-to-market (TTM) for the features of your e-commerce site. Achieving this goal would depend on:

- Simplifying and automating the processes.
- Delegating resource management to application owners (or domain owners) to enable them to manage the capacity of their resources within predefined limits.

- Defining pre-approved requests. This is an operational step that will enable the operations that are performed within a pre-defined framework to bypass most approval levels.
- The definition of a service catalog that exposes a self-service interface to the cloud users.

SLA-Driven Management

The management of IT services based on service level agreements (SLA) provides a way to describe contracts between the provider and the users. These contracts can be used to drive the behavior of the cloud computing infrastructure by defining multiple classes of service. The maximum efficiency of the cloud infrastructure is achieved when the resources are assigned to best-effort classes of service that can be preempted (or repurposed) to satisfy higher levels of service.

SLA along with chargeback can also be used as incentives to drive a specific behavior to cloud users.

Automation

Operation automation, especially the automation of routine tasks, is contributing to the infrastructure efficiency by:

- Reducing the human cost to manage the resources
- Decreasing the manual time to recovery (MTTR) by enabling automated remediation
- Increasing availability by reducing human errors
- Enabling efficient resource management though repurposing
- Capturing the tribal knowledge in well-documented workflows

Self-Healing

In addition to self-healing, a cloud computing infrastructure is more resilient to localized failures because of its level of automation and the focus on achieving the SLA. Because the allocation mechanism is dynamic, users will not (and should not) be concerned about where their workload is running. However, this information has to be available for cases where automation is not implemented. This will effectively eliminate traditional remediation methods. The cloud computing management system should address the remediation process, and also include the human approval, if your company requires it.

Service Orientation

Service orientation means to accomplish loose-coupling between application components to enable composition and facilitate rewiring of application components dynamically. Service orientation also enables quality of service (QoS) mechanisms that are implemented by the infrastructure (such as rate limitation), which in turn facilitate the resource management at the datacenter level. This property will not only apply to the applications that are deployed to the cloud infrastructure, but also mainly to the infrastructure services that are exposed, such as storage.

Multi-Tenancy

The majority of cloud computing infrastructures have multiple domains and tenants. Therefore it is critical to provide isolation between domains that have different requirements such as confidentiality and availability. Multi-tenancy should be considered at multiple levels:

- Role-based access control (RBAC) protecting access to sensitive functionalities
- Resource allocation policies that are aware of domain needs
- Information partitioning (logs, monitoring traces, and so on)

Cloud Security

It is essential to get the security in any distributed computing paradigm (such as cloud computing) correctly from the beginning because sharing the infrastructure and repurposing resources have the inherent potential to create security holes. Cloud computing infrastructure should take into account the security requirements of each application and provide secure isolation when deemed necessary. This isolation will impact the fluidity to some extent, as it creates partitions where resource sharing is prohibited or constrained. In this section we discuss the principles of cloud computing security from various aspects including the architecture review, implementation, operations, and governance.

Architecture Review

Architectural aspects of the cloud have to do with security aspects of information when:

- **In transit:** That is, information is exchanged between different hosts or between hosts and users.
- **At rest:** That is, when information is persisted in the storage media.

Information security aspects of any shared infrastructure are concerned with secure communication, authentication, and single sign-on (SSO) and delegation. Secure communication issues include those security concerns that arise during the communication between two entities. These include confidentiality and integrity issues. All data sent by users should be accessible to only legitimate receivers: This is referred to as confidentiality. All data received should only be modifiable by legitimate senders: This is referred to as integrity. There are also issues related to authentication, where the identities of entities involved in the overall process can be accurately asserted. These are critical issues in all areas of computing and communication and become exceedingly critical in cloud computing because of the heterogeneous and distributed nature of the entities involved in it. The main objective of the architecture review phase is to obtain a general understanding of the cloud and the applications running in it, and to identify obvious security flaws that might be present. A moderately-detailed diagram of the application architecture is a good starting point for this phase. Understanding the persistence and protection mechanism of personally identifiable information (PII) and any other security-sensitive information is one of the main objectives of this phase. The process does not aim to perform an architecture review of every single application that utilizes the cloud computing infrastructure. Suggested criteria are the popularity of the applications and their complexity.

Centralized Authentication

Using digital certificates is the cornerstone of strong authentication in the cloud infrastructure. Every user and service in the cloud is identified via a certificate, which contains information vital to identifying and authenticating the user or service. There are, however, alternative authentication methodologies, such as token-based and username/password authentication schemes. The choice of the authentication technique that you implement in your e-commerce solution is mainly mandated by ease of use by the users and services, and the level of protection that is needed to secure the infrastructure.

The service layer in the cloud computing infrastructure that is in charge of controlling the operation of the cloud (i.e. Control Plane) should provide an authentication and authorization framework that controls access to the management functions and sensitive infrastructure resources. Authorization is described later in this chapter.

Single Sign-On and Delegation

The orchestration of functions across multiple tools that compose the control plane requires that the access to said-tools be authenticated. The implementation

of a single sign-on (SSO) functionality that is integrated with these tools provides the best way to avoid proliferation of passwords and login methods both for system administrators as well as automation tools. The steps required to implement a single sign-on solution are:

- Defining centralized user and role repository for management tools
- Implementing an SSO framework that supports multiple access control methods (Web, Web services, API-based, and so on)
- Integrating the control plane tools with the SSO framework

For the users of the cloud computing infrastructure, it must provide SSO and delegation capabilities to reduce the number of times that the users must enter their credentials when multiple resources are used (a common use case in distributed management scenarios). This is done by creating a proxy that consists of a new certificate (with a new public key in it) and a new private key. The new certificate contains the owner's identity that is modified slightly to indicate that it is a proxy. The new certificate is signed by the owner rather than a certification authority (CA). The certificate also includes a time stamp after which the proxy should no longer be accepted by others.

Role-Based Access Control

The authorization framework should support Role-based access control (RBAC) as defined in the NIST RBAC standard[10] and implemented in the XACML[11] specification.

Credential Store

The repository that is used to store the credentials of tools and devices is an important asset. The automation of operations on managed elements such as routers and servers requires that the automation tool and the devices authenticate each other. The credentials used for the authentication have to be provided at the authentication time by the automation framework. Multiple methods can be used to provide these credentials:

- Store the credentials in the script/workflow.
- Ask the user for the device credentials.
- Use a secure centralized credential store.
- Use SSO solutions such as Kerberos.

The ideal situation where SSO is used may not be possible in all cases because devices do not support such a model. The first two methods are either not secure

or not possible for a fully automated model. This leaves us with the centralized credential store that can provide the required features:

- Programmatic access to credentials that are used in interactive protocols (for example, SSH)
- Centralized management of credentials
- RBAC for those credentials

Secure Communication and Storage

Public key cryptography (asymmetric cryptography) is the preferred technique for infrastructure security, both to address the protection of data at rest (encryption of data when persisted) as well as information in transit (SSL/TLS). In public key cryptography, the entities generate public/private key pairs based on cryptographically secure mathematical functions. When data is encrypted by the public key (either when persisted or when in transit) it can only be decrypted by the private key corresponding to the public key. The public keys are known to everyone.

Isolated Management

Management of the cloud computing infrastructure is a privileged operation. The network that is used by the control plane to communicate between its components or with the devices should be isolated from the production network. This implies that access to light-out management modules that enable hands-off management must be isolated from the data traffic from the time that the resources are wired to the network. Following this policy prevents many attacks and retains access to the management infrastructure in case of the data network failure.

Regulatory Compliance

Operational and regulatory compliance require that the infrastructure comply with the rules that are defined either by regulatory bodies or corporate rules. While more operational than technical, it is critical to design and operate your e-commerce cloud computing infrastructure with close attention to such compliance requirements.

Distributed Trust

The cloud computing infrastructure and the services it offers must be constructed in a dynamic fashion from components whose trust status is hard to determine. For instance, a user that trusts an entity may not necessarily trust the same entity

to delegate his rights further. Determining trust relations between participant entities in the presence of delegation is important, and delegation mechanisms must rely upon stringent trust requirements.

Freshness

Freshness is related to authentication and authorization and is important in many management applications, or more generally, shared services. Validity of a user's proof of authentication and authorization is an issue when user rights are delegated and where the duration of a job may span over a long period of time. Furthermore, some applications may want to state the number of times a given user or service may be allowed to access a resource. This is a nontrivial problem when one user's rights are delegated to another user that may thereafter wish to access the resource. Asserting the freshness of the right to perform an operation is a critical aspect of any cloud infrastructure.

Trust

Per Grandison and Sloman, trust is defined as "the firm belief in the competence of an entity to act dependably, securely, and reliably within a specified context." Because most complex applications can, and will, span multiple security domains, trust relationships between domains are of paramount importance. Participants in a distributed computing environment should be able to enter into trust relationships with users and other applications. In a distributed environment, trust is usually established through the exchange of credentials, either on a session or a request basis. Because of the dynamic nature of cloud computing environments, trust can be scarcely established prior to the execution of an operation. Further implementation details are provided in the "Trust Governance" section.

Secure Isolation

The term "isolation" refers to protecting the host data from outside workloads. A job or workload from a malicious user can corrupt local data, crash co-located workloads, and make the local system unusable. The isolation solutions aim to protect against this behavior. Two common techniques to establish isolation are *effective confinement* and *signed application*.

Effective Confinement

Effective confinement (or effective containment) is a type of isolation technique that keeps the *un-trusted* workloads in a protected (aka *sandboxed*) environment so

that even if the job is malicious, it remains confined to the isolated environment. This type of isolation can be achieved through several mechanisms:

- **Application-level sandboxing:** This mechanism, also known as Proof Carrying Code (PCC), enables the code provider to generate proofs of the safeness of the code and embed it inside the compiled code.
- **Virtualization:** As noted in Chapter 1, virtualization is a technique to allow the applications to run in isolated environments called *Virtual Machines* (VM).
- **Flexible kernels:** These systems typically include kernels, which can be extended in a flexible manner for better performance and isolation.
- **Sandboxing:** Perhaps the most popular of the isolation systems, these systems typically enforce isolation through interrupting system calls and loadable kernel modules.

Signed Application

Deploying a signed application is an effective way of establishing trust and deploying a verification mechanism to detect whether a trusted application is modified, which in turn could indicate malicious behavior. The process of signing an application is straight-forward. At a high-level it is:

1. Bundling together files of a relationship, such as the application binary and its associated configuration files
2. Obtaining a one-way hash of the bundle in Step 1 (fast operation)
3. Encrypting the one-way hash in Step 2 with the private key of the signer (cloud infrastructure in our case) to attest to the content corresponding to the digital envelope (includes other cryptographic artifacts)
4. Packaging the digital envelope created in Step 3 and the bundle in Step 1

The process of verification is also straight-forward.

1. Validate the integrity of the signature (assumes the availability of signer's public key) to prove the integrity of the one-way hash in the digital envelope.
2. Obtain the same one-way hash used for signing from the bundle.
3. Compare the calculated one-way hash with the one carried within a digital envelope.

It is important to note that a digital signature assumes the availability of signers' public keys (a sound assumption) and provides us with the verifiable-assertion that the claimed entity has attested to the integrity of the content. The latter property cannot be achieved if only a one-way hash is used.

Authorization

Another important security mechanism that must be implemented in a scalable way in cloud computing is authorization infrastructure. Similar to any other resource sharing systems, cloud computing requires resource-specific and system-specific authorizations. It is particularly important for systems where the resources are shared between multiple participants, and participant-wide resource usage patterns are predefined. Each participant can internally have user-specific resource authorization as well. The authorization systems can be divided into two categories: *virtual host level systems* and *resource level systems*. Virtual host (VH) level systems have centralized authorization mechanisms that provide credentials for the users to access the resources. Resource level authorization systems, on the other hand, allow the users to access the resources based on the credentials presented by the users.

- **Virtual host level:** VH level cloud authorization systems provide centralized authorization mechanisms for an entire VH. These types of systems are necessitated by the presence of a VH, which has a set of users, and several resource providers (RP) who own the resources to be used by the users of the VH. Whenever users want to access certain resources owned by an RP, they obtain a credential from the authorization system, which gives certain rights to the user. The user then presents the credentials to the resource to gain access to the resource. In this type of system, the resources hold the final right for allowing or denying users to access them.
- **Resource level:** Unlike the VH level authorization systems, which provide a consolidated authorization service for the VH, the resource level authorization systems implement the decision to authorize the access to a set of resources. Therefore, VH level and resource level authorization systems look at two different aspects of cloud computing authorization.
- **Revocation:** Revocation is an integral part of both authentication and authorization. Revocation is crucial for authentication in case of a compromised key, and for authorization when a participant is terminated, or a user's proof is compromised or otherwise untrustworthy. There are two mechanisms to implement revocation:
  - **Active Revocation Mechanism:** In this type of revocation, there is a communication between the user and the receiver access control mechanism, based on which the user is denied further access to the resource. This type of mechanism can operate very quickly and the revocation can happen as soon as the compromised identity is detected. In an X.509-based system, this can be done through the use of a certificate revocation list (CRL) issued by the authority, and the verifying authority or the access controller needs to check whether a CRL exists

for the credentials sent by the user. There are two types of overheads associated with such systems. There is an overhead of generating the CRLs to send to the access controller. However, the more significant overhead is that each time the access controller performs an access check it needs to see whether there is a CRL associated with each user credential. This may lead to a loss of scalability, especially if there are a huge number of users.

- **Passive Revocation Mechanism:** This type of revocation mechanism is accomplished through expiration times provided in most certificates. During the generation of certificates, an expiration time is provided after which the certificate is deemed invalid. In terms of scalability, the passive revocation mechanisms are better than their active counterparts. However, the scalability comes at a cost. Let's assume that the certificate is issued at time T and expires at time T + t. Now let's assume that the user is compromised just after T. Then for a period of t, the adversary is capable of compromising the system further. If t is small, then the system is more secure. However, smaller t also indicates that more authorizations are required, reducing the scalability of the system. Therefore, there is a trade-off between the scalability and security, which is tuned by the choice of the time t.

Threats

One school of thought defines security purely based on the types and categories of threats it should protect against. One of the most important security threats existing in any infrastructure is the malicious service disruption created by adversaries. Many examples exist in the Internet space where servers and networks are brought down by a huge amount of network traffic, and users are denied access to a service. Because cloud computing deployment has not reached the "critical mass" yet (that is, not every single service provider on the Internet is deploying cloud computing), the service level attacks are still limited. However, with cloud computing poised for a huge growth in the next few years, this area is of the utmost concern. The cloud service level security issues can be further subdivided into two main types: *denial of service* (*DoS*) and *Distributed Denial of Service* (*DDoS*) issues; and *Quality of Service* (*QoS*) violation issues, which in turn is the forced QoS violation by the adversary through congestion, delaying or dropping packets, or through resource hacking.. The D/DoS is the more dangerous of the two where access to a certain service is denied.

DoS

The solutions proposed for DoS (denial of service) attacks can be categorized into two primary types: *preventive* solutions and *reactive* solutions. Preventive

solutions try to prevent the attack from taking place by taking precautionary measures. Reactive solutions, on the other hand, react to a DoS attack and are generally used to trace the source of the attack. Some examples of preventive solutions are filtering, throttling, location hiding, and intrusion detection. Examples of reactive solutions include logging, packet marking, and link testing.

QoS

This is an active area of research and several architectures and solutions have been proposed. Most of these solutions rely on some amount of monitoring and metering systems, which try to detect the QoS (Quality of Service) levels of the system and then make decisions to raise the alarms.

Applications

Enumerating a complete application threat model is practically impossible as it requires enumerating all the possible attack vectors, which in turn results in a combinatorial explosion of the system-state space. Delineating the *exact* risk associated with each threat is similarly meaningless: This is analogous to an insurance agent saying "You need to tell me exactly when the next earthquake hits, and identify which buildings will be damaged, and precisely what kind of damage so I could give you an insurance quote." Since that's not possible for obvious reasons, the best next thing to do is to perform statistical approximation of the most popular attacks and protect against the top items. The following is a list of most common Web application attacks that the cloud computing infrastructures should protect against. This is by no means a comprehensive list, as the attack field is dynamically evolving:

- Cross Site Scripting (XSS)
- Cross Site Request Forgery (CSRF)
- Insufficient authorization
- Information leakage
- HTTP response splitting
- Content spoofing
- Predictable resource location
- Open redirects
- Brute force
- Abuse of functionality
- Session fixation
- Directory indexing

In the context of cloud computing, for the applications whose intent is not to be malicious the, top threats appear to be the following. The list of attack vectors is not ranked in any specific order:

- **XSS (Cross Site Scripting):** Starting point for many of the attacks listed here. XSS vulnerabilities can be exploited to inject whatever code is desired. Proper input validation would prevent this.
- **CSRF (Cross Site Request Forgery):** Failing to ensure that apps are immune to CSRF-based attacks could leave the cloud computing application and its users wide open to "drive-by" attacks that make use of whatever functionality exists in the particular application (which may be quite elevated) all without the users' awareness. Sending a randomized user-specific token with every request for the app to check would help prevent CSRF. However, any application that is vulnerable to XSS is also vulnerable to CSRF because one can glean the mentioned token via XSS.
- **Open redirects:** If an application were vulnerable to XSS-based attacks, redirecting the user's browser (to a spoofed sign-in page, for example) would be simple if the app is vulnerable to persistent XSS, and a bit more difficult with reflected XSS (the user would have to visit a specially crafted URL that contains the redirection payload in it).
- **Malware drops/drive-bys:** Leveraging an XSS vulnerability, malicious code can be forced upon a user and exploit a vulnerability in the browser, browser plug-in, and so on to drop a piece of malware on the user's machine, and without the user's awareness. iFrame attacks (aka *iFrame hijacking*) are only one mechanism to mount this exploit and are a vector of choice for attackers. Malware is wrapped in JavaScript, gets past the defenses, unpacks through the browser, escapes the iFrame-jailhouse, and compromises the system without anything knowing it went by.
- **Malicious links:** Using XSS, a malicious user could embed a link (i.e. `<a href="...">`) to a malicious site that executes any of the previously mentioned attacks.
- **Third-party availability:** If some parts of an application reside off the protected portion of your cloud computing infrastructure, or leverage off-infrastructure resources, there will then be reliance on the availability/integrity of the third party's infrastructure.

It must be noted that most of the attacks in the preceding list exploit holes in the applications themselves. In other words, the application is not meant to be malicious. These same attacks would be *significantly* easier to execute if the developer of the application *intended* for the application to be malicious, of course, and are a very real possibility if an untrusted developer is allowed to write applications for your cloud computing and e-commerce infrastructure.

Operational Aspects

The operations aspect of cloud computing security should address its nodes (hosts) and its communication network.

Host-Level Security

Host-level security issues are those that make a host apprehensive about affiliating itself to the cloud computing system. The main issue here is data protection. Whenever a host is affiliated to the infrastructure, one of the chief concerns is the protection of an already-existing data in the host. The concern stems from the fact that the host submitting the job may be untrusted or unknown to the host running the job. To the host running the job, the job may well be a virus or a worm that can destroy the system. This is called the *data protection issue.*

To address the data protection issue, you need to implement isolation so that access to data is restricted to the cloud computing or external applications. As noted in the "Effective Confinement" section, various isolation techniques exist to accomplish data protection.

Network

In the context of cloud computing, network security issues assume significant importance primarily because of the heterogeneity and high-speed requirements of many of the applications running in it. Moreover, the infrastructure inherits some of the generic network issues as well. Access control and isolation are important requirements for traffic flowing through the networks. In this area, integration with virtual private networks (VPN) and firewall technologies also gain significance. Routing of packets in networks based on routing tables is a specific network issue. Attacks in routing include link and router attacks, which may cause significant destruction. Many of the issues still require further research. Multicasting is an efficient means of information dissemination and is also an important factor for cloud computing infrastructure security. Participant authentication, key management, and source authentication are specific security issues in multicasting. Another topic of interest is the integration of *sensor networks*. Several sensor network attacks, such as sybil attacks, wormhole and sinkhole attacks, and node hijacking, will need to be tackled before the sensor-cloud vision can be realized. Below is the list of some practical security issues in high performance interconnects:

- **MAC (Mandatory Access Control):** Many of the Web services solutions cannot work effectively with firewalls and VPN solutions, which have become ubiquitous in today's enterprises. The area requires significant additional research.
- **Routing:** The routing technologies available today are inherited from traditional networking. Most routing protocols use digital signatures and

passwords for message exchange, which do not counter the advanced attacks such as source misbehavior. Inconsistency detection is an attractive technique that is available today and can be deployed.

- **Vetting:** Vetting in the context of cloud computing applies to the following categories:
  - **Vendors:** This item applies *only* if a third-party entity acts as the provider of infrastructure. The main focus of this phase from the security perspective is to understand and identify security risks of third-party networks and their operations.
  - **Operations staff:** This phase describes the vetting process for privileged users within cloud computing infrastructure who make critical-operation decisions such as changing capacity; modifying host, network, or application configuration; utilizing corporate resources; and accessing sensitive audit trails and logging information.
  - **Applications:** Vetting the applications deployed to cloud computing infrastructure is a non-trivial task and has to be undertaken very carefully, as the process is cumbersome and tends to lack scalability.
  - **Penetration testing:** While your security staff might want to perform penetration testing and code audit for all applications and infrastructure components, the objective is to ensure that this step is cost-effective and scalable. In practice, this will be required only for the most complex applications that extensively utilize infrastructure services.

Please note that the preceding items all suffer from snapshot syndrome; that is, when something in the application or the infrastructure changes (package update, adding new functionality, and so on) after the audit and vetting is successfully completed, the vetting results don't apply and are considered void.

Governance

Governance is important for distributed computing, or any shared infrastructure such as cloud computing for that matter, because the execution environment is usually heterogeneous in nature and consists of multiple entities, components, users, domains, policies, and stake holders. The different governance issues that administrators are worried about are credential and trust governance, as well as MLT (Monitoring, Logging, and Tracing) issues.

Credential Governance

Governance of the *credentials* is a critical aspect of cloud computing infrastructure because there are many systems that interact with each other and require different sets of credentials for accessing them. Credential governance systems

store and manage the credentials for a variety of systems and users who can access them according to their needs. This mandates specific requirements for the credential governance systems. For typical distributed management credential governance systems, mechanisms should be provided to obtain the initial credentials. This is called the *initiation requirement*. Similarly, secure and safe *storage* of credentials is equally important. Additionally, the credential governance systems should be able to access and renew the credentials based on the demand of the users. A few other important requirements are *translation*, *delegation*, and *control* of the credentials. Considering the preceding requirements, credential governance systems are mainly of two types: *credential repositories* or credential storage systems, and *credential federation systems* or credential share systems. The first set of systems is responsible for storing credentials while the second set of systems is responsible for sharing credentials across multiple systems or domains.

Repositories

The basic purpose of credential repositories is to move the responsibilities of credential storage from the user to these systems. Examples of credential repositories include smart cards, virtual smart cards, and MyProxy Online Credential Repositories (`http://grid.ncsa.illinois.edu/myproxy`). Smart cards are credit card–sized tokens that contain a user's secret key material. Virtual smart cards embed the features of smart cards in the software. MyProxy is a popular implementation of credential repositories specifically for Grid and cloud computing systems.

Federation

Credential Federation systems, protocols, and standards are used for managing credentials across multiple systems, domains, and realms. Examples in this space include VCMan, which is Grid-specific, and Community Authorization Service (CAS), which offers interoperability across multiple domains. KX.509 is a protocol that provides interoperability between X.509 and Kerberos systems. A standard called the Liberty Framework was developed by the Liberty Alliance (`www.projectliberty.org/`), which was a consortium of about 150 companies for creating and managing federated identities. This project has now moved to the Kantara Initiative (`http://kantarainitiative.org/`). Another popular open source solution in this space is Shibboleth (`http://shibboleth.internet2.edu/`).

Trust Governance

Governing the trust is one of the most sensitive aspects of cloud computing infrastructure. Trust is a multifaceted entity that depends on a host of different ingredients, such as the reputation of an entity, its policies, and opinions about that entity. Governing trust is crucial in a dynamic infrastructure where hosts and users constantly join and leave the system. Therefore, there must be

well-defined mechanisms to understand and manage the trust levels of systems and new hosts that join the infrastructure. The trust life cycle is mainly composed of three different phases: *trust establishment*, *trust negotiation*, and *trust management*:

- **Trust establishment:** The trust establishment phase is generally done before any trusted group is formed, and it includes mechanisms to develop trust functions and trust policies.
- **Trust negotiation:** The trust negotiation phase is activated when a new un-trusted system joins the current distributed system or group.
- **Trust management:** Trust management is responsible for recalculating the trust values based on the transaction information, distribution or exchange of trust-related information, and finally updating and storing the trust information in a centralized or in a distributed manner.

The main characteristics of trust governance systems are scalability, reliability, and security. In other words, the trust governance systems should scale in terms of message, storage, and computational overheads. Trust governance solutions should be reliable in the face of failures and should also be secure against masquerade, collusion, and Sybil[12] attacks. Trust governance systems can be divided into reputation-based and policy-based categories:

- **Reputation-based:** This category of systems operates based on trust metrics that are derived from local and global reputation of an entity. Example solutions include PeerTrust, XenoTrust, and NICE.
- **Policy-based:** In policy-based systems, the different system entities exchange and govern credentials to establish the trust relationships based on predefined policies. The primary goal of policy-based systems is to enable access control by verifying credentials and restricting access to credentials based on policies. These systems usually have a policy-based trust language. Examples include PeerTrust and TrustBuilder.

Monitoring, Logging, Tracing

MLT (Monitoring, Logging, and Tracing) is the third and one of the most crucial components of governance. Establishing an efficient MLT is essential in cloud computing for two reasons:

- Different consumers can be charged based on their usage (Monitoring).
- Resource-related information can be logged for auditing or compliance purposes (Logging & Tracing).

MLT operates at application, system, and infrastructure levels. The MLT governance infrastructure should be configurable to allow the degree to which a selected set of applications and hosts are monitored, logged, and traced.

Furthermore, there should be a configurable *infrastructure level* MLT to ensure the health of the system as a whole. Infrastructure level monitoring systems are much more flexible than other monitoring systems and can be deployed on top of other monitoring systems. Many of the infrastructure level monitoring systems provide standards-based interfaces for interacting, querying, and displaying information in standard formats.

At each of the levels mentioned, there are different stages of MLT: *data collection*, *data processing*, *data transmission*, *data storage*, and *data presentation*. The data collection stage involves collecting data through different sensors located at different collection points. The gathered data can be static in nature (such as network topology and machine configuration) or dynamic (such as CPU utilization, memory consumption, system load, and so on). The data processing stage processes and filters the data based on different policies and criteria from the data collected from the sensors. The data transmission stage involves the transmission of collected and processed data to different interested entities within the system. Transmission involves sending the data over a medium in a format that is understood by other parties. There may be a need for storage of gathered or processed data for future references, which in turn is carried out in the data storage stage (Tracing). Finally, the data presentation stage presents the data in a format that is understood by interested entities within the system.

Summary

We started this chapter by discussing the basics of commerce (although as noted in the introduction, this wasn't exactly a history chapter, was it?). We then explained the payment systems and the technical mechanics of money movement, and worked our way up to distributed computing and its most scalable paradigm: cloud computing. In the last part, we explained important characteristics of cloud computing and how to implement them correctly. In the next chapter, you will learn more details about how to make the system scalable (the important "-ilities") and then continue to delve deeper into the realm of security. Hang on tight; the fun is just about to start.

Notes

1. There's also a third meaning for the term "commerce" per Merriam Webster, but that meaning does not exactly bode well with the scope of this book.
2. `http://en.wikipedia.org/wiki/History_of_banking`
3. `http://en.wikipedia.org/wiki/Credit_card`
4. Forrester Report on Mobile Payment.
5. Orfali, R. et al., *The Essential Client/Server Survival Guide*. Wiley 1997. p13–14.

6. Chakrabarti, A. *Grid Computing Security*, Springer 2007.
7. www.yousendit.com
8. Stoica, I., et al.,"A Scalable Peer-to-peer Lookup Service for Internet Applications," in *Proc. ACM SIGCOMM,* San Diego, 2001.
9. Ratnasamy, S., *A Scalable Content Addressable network*, *Ph.D. Thesis*, University of California, Berkeley, 2002.
10. http://csrc.nist.gov/rbac/
11. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf
12. Sybil attacks are the class of attacks where a reputation system is subverted by forging identities within a P2P network.